(m,k)-WFQ : Integrating (m,k)-Firm Real-Time Constraints into Guaranteed-Rate Networks

Anis Koubaa, Ye-Qiong Song

(m,k)-WFQ: Integrating (m,k)-Firm Real-Time Constraints into Guaranteed-Rate Networks

Anis KOUBAA, Ye-Qiong SONG
LORIA-INPL-UHP Nancy I
516, Rue du Jardin Botanique 54506 Villers Les Nancy
{akoubaa, song}@loria.fr

Abstract

Guaranteed-Rate (GR) servers, such as Weighted Fair Queueing (WFQ) and its variants, have been widely used to give mainly bandwidth guarantees and consequently delay guarantees for real-time flows provided that their arrivals are upper-bounded. Problems may arise if a bursty traffic with a small service share needs a specific short delay. In fact, the higher the service share is, the lower the delay the flow gets. However, WFQ and its variants are share-driven servers and no temporal constraint is considered in the scheduling process. Therefore, having in mind that real-time streams could tolerate some deadline misses according to their (m,k)-firm constraints, we propose a new scheduling technique called (m,k)-WFQ that extends WFQ to also consider (m,k)-firm temporal requirement. Analytic expressions using Network Calculus theory are derived to give deterministic upper bound on delay provided by the (m,k)-WFQ scheduler. Theoretical Results and simulations show lower average and maximum delays provided by the proposed scheduling algorithm, without much degrading bandwidth fairness.

Contents

Keywords

WFQ, (m,k)-firm, Network Calculus, Delay Guarantees, Bandwidth Guarantee

1. Introduction

Real-time applications are of growing use over the Internet and a lot of work has been developed to provide quality of service (QoS) for real-time applications. Main scheduling approaches use Guaranteed-Rate (GR) servers, such as WFQ, to reserve bandwidth for real-time streams and make a bounded end-to-end delay for each served stream. However, this delay bound is affected by the bursty nature of the flow. Actually, for a given bandwidth share, when the burst size is important, the delay of real-time packets gets large and the temporal QoS may be transgressed [11]. A naïve solution may consist in increasing the reserved bandwidth for the low share bursty stream to meet its deadline requirement. However, this technique violates the throughput fairness and misuses the bandwidth resources. This limit comes from the selection criterion in WFQ server. In fact, packets are selected in increasing order of their virtual finish time [12]. This tag only depends on the service share and the packet length of the flow. Flows with high service share are served faster. Therefore, we notice that WFQ doesn't take into account temporal constraints of served flows. Increasing the service share to meet deadline requirement is not efficient, since it leads to underutilize bandwidth resources.

To smooth this problem, Wang et al. proposed in [2] a technique called PWFQ that combines fixed-priority assignment and WFQ scheduling algorithm in order to better manage the delay bounds for various flow sessions. The problem to resolve was how to provide short delay guarantees for flows with low service share. The outcome of this method is to reduce the upper bounds on delay for flows with low service-share by assigning them higher priorities within a *sliding window*. A window defines a (virtual) time interval in which all packets whose virtual finish times fall into are considered to have a similar finish time. The server selects the highest priority packet to be served within the sliding window. This technique leads to decouple the delay from the service share and provides better delay guarantees for low share flows without much degrading the delay of other flow sessions, but the choice of the optimal window size might be complex.

Moreover, when serving several bursty streams, the server may suffer from congestion if the system queue approaches its capacity limit. This situation occurs when incoming bursts are significant. Predictive mechanisms are used to prevent congestions like for example the RED algorithm [1], which discards randomly each arriving packet with a

certain probability when the average queue size exceeds a preset threshold. Such behavior could affect a real-time connection if packet drops are made inadequately.

To alleviate these problems, we introduce in this paper a fair scheduling technique for providing (m,k)-firm real-time guarantees proposed in [3,4]. The idea takes profit from the tolerance of occasional deadline misses according to a (m,k)-firm pattern to selectively discard some packets without violating the overall performance of the application when the flow is served by a WFQ scheduler. Hamdaoui and Ramanathan mentioned in [3] that some real-time applications, such as multimedia flows and embedded systems, could tolerate some deadline misses if they occurred occasionally and not successively. Moreover, The study on the MPEG video flow presented in [14] stated that the effect of the packet loss on the QoS of the application depends on when and how the loss occurs. For example, according to the compression mechanism of MPEG streams, the loss of data in I (Intra-coded) and P (Predicted) frames of an MPEG flow, will propagate and cause errors in later frames until a new I frame arrives, where as the B-frame (Bi-directional coded frame) loss has no propagation effect. Therefore, it is more efficient to guarantee the I and P frames on time delivery, if possible, to make the decode process more effective.

In general, when packet-drops are adequately made according to a well-defined pattern, it reduces the system load, which results in speeding-up packet transmission by reducing the queueing time. This approach enhances the delay bound by smoothing the bursts of served sessions. Simulation, as well as analysis, shows noticeable improvements on the delay guarantees, throughputs and system utilization, making proof of the efficiency of the proposed algorithm.

More specifically, a flow is said to have (m,k)-firm requirement if m out of any k consecutive packets must meet their respective deadlines where $m \leq k$. This kind of requirement is a particular case of weakly hard real-time constraints detailed in [4] which is useful to represent the tolerance of some deadline misses. For example, a hard real-time flow could be considered as (m,k)-firm flow when $m=k$.

The motivation of this work is to integrate the temporal constraints, namely (m,k)-firm requirements, in the GR servers to provide not only bandwidth guarantees but also to satisfy the real-time requirements.

The proposed algorithm is also a GR server that takes into account (m,k)-firm constraints according to a pre-defined pattern. We propose an

extension on WFQ packet selection function to meet the (m,k)-firm requirement of each served stream. However, (m,k)-firm constraint, which represents a deadline-driven parameter, and WFQ scheduling algorithm, which represents a share-driven parameter, are not directly related. Therefore, we describe through this paper how to integrate (m,k)-firm constraints into GR server to improve delay guarantees.

For a deterministic (m,k)-firm guarantee, we provide analytic results to evaluate upper bound on delays guaranteed by the (m,k)-WFQ scheduler. For this purpose, we introduce the new concept of *(m,k)-filter*, defined as a filtering component in Network Calculus, that shapes the arrival curve of a stream according to its (m,k)-firm constraint.

The rest of this paper is organized as follows. Section 2 gives an overview of the (m,k)-firm concept. The proposed (m,k)-WFQ algorithm is presented in section 3. Section 4 introduces the (m,k)-filtering concept to adapt Network Calculus framework to the use of (m,k)-firm constraints and presents the analytic delay evaluation . Simulation study and results are deeply discussed in section 5. Section 6 concludes the paper.

2. Overview of (m,k)-Firm Guarantees

This section explains how the (m,k)-firm timing constraint could be useful for a real-time stream to express its deadline miss profile.

A stream is said to have (m,k)-firm requirement if at least m packets inside any window of k consecutive packets must meet their required deadline. If more than k-m deadline misses occur in a specified window, the stream temporal constraint is transgressed, or is said to be in *dynamic failure state*.

We define the concept of κ-*pattern* that specifies the organization of deadline miss and meet within a window of k packets.

Definition 1.

The κ-pattern of a stream having (m,k)-firm deadline requirement is the succession of k elements from the alphabet $\Delta = \{O, M\}$ *where:*

$$\begin{cases} O & \text{Stands for an Optional packet} \\ M & \text{Stands for a Mandatory packet} \end{cases}$$

and contains exactly m 'M' symbols.

$\kappa(i)$ denotes the i^{th} element of the κ-pattern for $1 \le i \le k$

Using κ-patterns, the stream's packets are divided into *optional* and *mandatory* parts. To satisfy the (m,k)-firm constraint of a flow, it is sufficient that all mandatory packets meet their deadlines[1]. Optional packets could be dropped or transmitted depending on the scheduling algorithm. The miss of all or some optional packets would not affect the temporal QoS of the stream. Hence, a *Firm-Guaranteed service* consists in guaranteeing the deadline meet of all mandatory packets.

For example, a stream having (3,5)–firm constraint could satisfy several κ-patterns ('*MOOMM*', '*MMOMO*', …) of length 5 and having exactly 3 mandatory packets. Based on the κ-pattern, the classification of the whole packets of the stream follows *Proposition 1*.

Proposition 1.

Consider a stream with (m,k)-firm constraint.

The n^{th} packet is classified as mandatory if and only if

$$\kappa(n \bmod k) = 'M' \qquad n=1,2,... \qquad \textbf{(1)}$$

where mod is modulus operator (remainder after division)

This technique may be used in different contexts. In [5], the classification into mandatory and optional parts has been used to enhance the Rate Monotonic scheduling process. Also in multimedia, this concept could be applied to select mandatory frames from a group of picture (GOP) using the MPEG compression standard. For instance, if a stream has a GOP structure IBBPBBPB, it could be considered as a (3,8)-firm flow and assigned a κ-pattern such $\kappa = 'MOOMOOMO'$. This fact says that all B-frames are considered as optional. Scheduler should take more care of I and P frames since they are mandatory.

Also, for an audio stream, that can tolerate no more than one deadline miss could be assigned a (2,3)-firm guarantee with a κ-pattern $\kappa = 'MMO'$.

In general, the specification of a κ-pattern for a flow is user-initiated or application-initiated. This is simply a way among several to classify packets according to (m,k)-firm constraints. This classification is static. However, we could imagine a dynamic classification made by the server according to a given algorithm. Dynamic classification is out of the scope of this current study.

---

[1] Notice that the inverse is not necessary true since in general the *m* "M" symbols can be arbitrary distributed among the *k* consecutive packets.

3. (m,k)-Weighted Fair Queueing

To make WFQ compatible with (m,k)-firm constraints, we integrate the concept of the κ-pattern into WFQ scheduler.

Let us consider a WFQ scheduling process that serves N streams $\{S_1, S_2, \ldots, S_N\}$ with constant service rate C and flow service share Φ_i. Each stream has its (m_i,k_i)-firm constraint.

The standard WFQ scheduling algorithm is based on the computation of the virtual finish time to emulate the fluid GPS system [6, 12].

The virtual finish tag of a packet is defined as:

$$F_i^k = \max\left\{F_i^{k-1}, V(t)\right\} + \frac{L_i^k}{\Phi_i} \quad \textbf{(2)}$$

where F_i^k is the virtual finish time of k^{th} packet of stream S_i. $V(t)$ is the virtual time when k^{th} packet arrives and L_i^k is the k^{th} packet size.

The standard WFQ scheduler selects the packet with the lowest finish tag. This tag does not consider any temporal constraint. It only depends on the sharing factor and the packet length.

However, our proposed solution, called (m,k)-WFQ, uses (m,k)-firm constraints by taking into account packet classification as *optional* and *mandatory* parts according to the corresponding κ-pattern. The mandatory part **must** be served and **should** meet the specified deadline. The optional packet is dropped unless it meets the specified deadline, *i.e.*, before serving an optional packet, the server checks if it will or not meet the required deadline. If it is the case, the packet will be served, otherwise, it will be dropped.

Therefore, the scheduling algorithm is divided into

- Priority assignment process,
- Service process.

Table 1 shows the (m,k)-WFQ scheduling algorithm.

The *priority assignment process* stamps each incoming packet with two tags: The *Finish Tag* as shown by equation (2), and *Priority tag* specifying if the packet is Optional or Mandatory according to the pre-defined κ-pattern of the served flow (*c.f.* equation (1)). Once the arrived packet is tagged, it is queued waiting for its service. It is also possible, if the application is aware of its (m,k)-firm constraint, that the packet is tagged by the source as optional or mandatory according to its κ-pattern.

Table 1. (m,k)-WFQ Scheduling Algorithm

```
Input
Streams Si = {(Period or Rate), Deadline requirement,
(m,k), (Jitter or Burst), Packet-Size)
Priority Assignment
At a^th packet arrival time of stream [i] {
   if ( κ((a mod k_i)+1) = 'M' ) then {
         Tag the packet as Mandatory 'M';
         }
   else  {
          Tag the packet as Optional 'O';
         }
   Compute Virtual Finish Time F_i^k ;
   Tag The packet with F_i^k ;
   Put the packet in the corresponding queue;
}
Service Discipline
server = idle;
While (buffer not empty) do {
   if (server!= busy) {
      if (there is at least one mandatory
          packet at the head of queue) then
             Select mandatory packet with
             lowest finish tag
      else   Select Optional packet with
             lowest finish tag;
      if  (packet is mandatory) then {
         send the packet;
         server = busy;
         }
       else { //Optional Packet
            if (deadline would be missed){
                  drop the packet;
                  server = idle;
                  }
             else  {
                  send the packet;
                  server = busy;
                  }
         }//end else 'Optional Packet'
      } //end if server!=busy
   if (server== busy) {
         wait until packet is transmitted;
         server=idle;
         }
}//end while
```

The *service process* is activated whenever there is, at least, a packet waiting in the buffer. First, the scheduler selects the packet with *lowest Finish Tag* among **only mandatory** packets from the head of active queues. If no mandatory packet is available, the selection is made among optional ones. If the selected packet is mandatory, it is immediately transmitted. However, for an optional packet, the scheduler predicts whether the selected packet would meet or miss its ***required deadline***

after being served. In the case of deadline meet the packet is served, otherwise, it is discarded and another packet would be selected.

Hence, the gain of (m,k)-WFQ is that it almost performs the same bandwidth guarantee as WFQ does. Furthermore the former reduces the delay when using (m,k)-firm temporal constraints by dropping optional packets. This fact is interesting in overload condition management since this drop technique is more suitable for real-time streams than the predictive RED algorithm. In such cases, dropping optional packets that are missing their deadlines provides lower delays for waiting mandatory packets. However, using standard WFQ, the delay bound would be more important due to the sharp effect on the incoming burst. Therefore, the benefit of the (m,k)-firm guarantee is to smooth this unwanted effect when the server is loaded.

A particle advantage of this scheduling algorithm is its simplicity and also, it does not introduce heavy computation overheads, so that its implementation remains feasible and easy to perform. Moreover, this mechanism could be extended to be used by an admission control function to accept streams according not only to their traffic specification (*Tspec* in IntServ QoS model), but also their (m,k)-firm requirements.

4. Delay Bound Analysis

In this section we present the analysis of the guaranteed delay bound by a (m,k)-WFQ scheduler using Network Calculus Formalism. For this purpose we introduce the concept of (m,k)-filtering to adapt Network Calculus to the use of (m,k)-firm constraints.

4.1 (m,k)-Filtering Theory

Assume that we have a flow with cumulative arrival function *R(t)* that has a (m,k)-firm deadline requirement. Let κ be the κ-pattern of this flow, as explained in section 2.

Definition 2. (m,k)-Filter device

We define an (m,k)-filter as a device, that for an arrival function R(t), makes the output $\tilde{R}(t)$ *where only mandatory packets of the corresponding flow are sent according to its* κ*-pattern. Optional packets are discarded.*

In this paragraph, we assume that each flow has a constant packet size denoted by *L*. The case of variable packet size is presented in the next paragraph.

Note that $R(t)$ denotes the input in terms of number of packets. To have the number of bits, we just need to multiply this quantity by the packet size *L*.

At once, if $m = k$, the output is exactly equal to the input. For $m < k$, $\widetilde{R}(t)$ is the cumulative number of mandatory packets within the interval $[0,t]$.

The following theorem expresses the arrival curve at the output of a (m,k)-filter.

Theorem 1.

Consider a constant packet size flow with a cumulative arrival function R(t), a κ-pattern κ and crosses an (m,k)-filter. $\widetilde{R}(t)$ *is the output of the (m,k)-filter if and only if,*

$$\widetilde{R}(t) = m \cdot \left\lfloor \frac{R(t)}{k} \right\rfloor + \Pi\left(R(t)\right)$$

with

$$\Pi\left(R(t)\right) = \sum_{n=\left(\left\lfloor \frac{R(t)}{k} \right\rfloor k\right)+1}^{R(t)} \overline{\kappa}(n \bmod k)$$

and $\forall 1 \le i \le k$, ***if*** $\kappa(i) = 'M'$ ***then*** $\overline{\kappa}(i) = 1$ ***else*** $\overline{\kappa}(i) = 0$

The proof is detailed in Appendix 1.

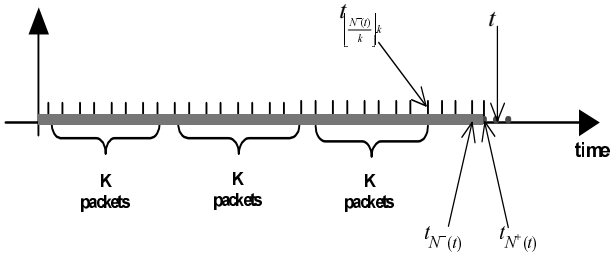
4.1.1 Case of Variable Packet Size

For a variable packet size flow that crosses an (m,k)-filter, definition 2 remains valid. However, Theorem 1 is no longer appropriate in this case, as mentioned earlier, since the number of packets that crosses the (m,k)-filter does not reflect the number of mandatory bits because the packet size is variable.

We propose to use the fluid model to derive the number of bits that crosses a (m,k)-filter for a flow with a variable packet size.

Note that $R(t)$ denotes the input in terms of number of bits. For this model, we denote by t_n the arrival time of the first bit of the n^{th} packet with length L_n. $N^-(t)$ denotes the number of packets fully received up to time t. $N^+(t)$ is the number of received packet, which include the current

one. We denote by C the server capacity. The n^{th} packet is completely received within a latency $\frac{L_n}{C}$.



Theorem 2 gives the arrival curve at the output of a (m,k)-filter for a variable packet size (fluid model).

Theorem 2.

Consider a flow with a cumulative arrival function R(t), a κ-pattern κ and crosses a (m,k)-filter. $\tilde{R}(t)$ *is the output of the (m,k)-filter if and only if,*

$$\tilde{R}(t) = \Lambda(t) + \overline{\kappa}\left(N^{+}(t) \bmod k\right).\left(R(t) - \sum_{n=1}^{N^{-}(t)} L_n\right)$$

$$\text{and} \quad \Lambda(t) = \sum_{n=1}^{N^{-}(t)} \overline{\kappa}\left(n \bmod k\right).L_n$$

The proof is detailed in Appendix 2.

4.1.2 Application to a Leaky Bucket Shaped Stream

Let us consider a constant-packet size stream with cumulative arrival function $R(t)$ shaped with a leaky bucket controller with σ is the maximum burst size and ρ is the average long-term rate. Then, the arrival curve at the output of the (m,k)-filter is given by theorem 3. (*cf.* figure 1).

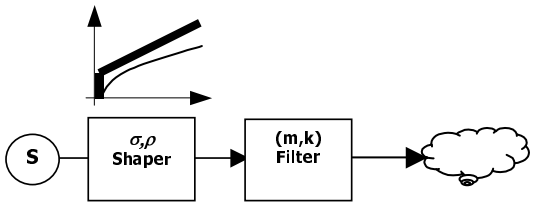


Fig 1. System Description

Theorem 3.

Consider a constant packet size stream S with arrival function R(t) upper constrained by the arrival curve $\alpha(t) = \sigma + \rho.t$ *and crosses a (m,k)-filter device. The output produced by the (m,k)-filter is bounded by the arrival curve* $\widetilde{\alpha}(t) = \widetilde{\sigma} + \widetilde{\rho} \cdot t$ *where:*

$$\begin{cases} \widetilde{\sigma} = \dfrac{m}{k} \cdot \sigma \\ \widetilde{\rho} = \dfrac{m}{k} \cdot \rho \end{cases}$$

and $t \in \mathrm{T} = \{t_0, t_k, t_{2k}, ..., t_{nk}, ...\}$ *where* t_{nk} *is* $(nk)^{th}$ *packet arrival time.*

We call this curve as the ***minimal arrival curve*** *of the stream.*

For the proof, refer to Appendix 3.

Figure 2 shows an example of stream with constant packet size $L = 2$ and a period $P = 1$ affected with a jitter uniformly distributed in [-0.45, 0.45]. According to [6,7], the stream is constrained by the arrival curve with a maximum burst size $\sigma = 3.8$ and an average long-term rate $\rho = 2$ (highest curve).

If the stream crosses a (3,5)-filtering device, the arrival curve of mandatory packets is reduced to a maximum burst size $\widetilde{\sigma} = 2.28$ and an average rate $\widetilde{\rho} = 1.2$. In this example, the κ-pattern is $\kappa = \{MOOMM\}$ as it could be observed from the figure.

We have introduced up to this section the basic background to deal with the upper bound on the delay guaranteed by a (m,k)-WFQ scheduling algorithm. Next section presents the delay bound analysis for a given (σ, ρ)-shaped flow served by a (m,k)-WFQ scheduler.

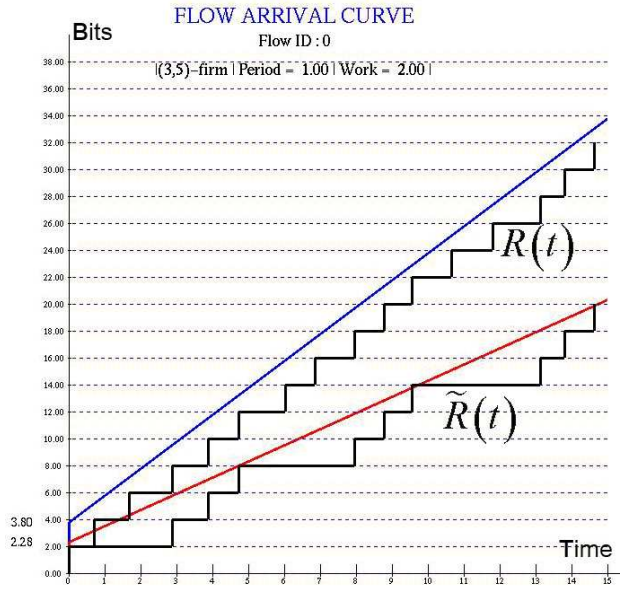


Fig 2. Example of Arrival curve

4.2 Delay Bound Analysis

In this paragraph, we formally examine the delay bound guaranteed for a flow served by a (m,k)-WFQ node and bounded by the arrival curve $\alpha(t) = \sigma + \rho \cdot t$. We assume that each stream has a constant packet-size. We extend the delay bound analysis of standard WFQ scheduler, detailed in [6], to take into account the (m,k)-firm constraints.

Table 2 presents the system analysis notations.

Table 2. Notations

C	Server capacity
Φ_i	Service share of i^{th} flow
L_{max}	Maximum packet size among all flows
σ_i	Burst size of i^{th} flow
ρ_i	Long-term average rate of i^{th} flow
$D_{i,\max}$	Maximum delay under WFQ server
$D^*_{i,\max}$	Maximum delay under (m,k)-WFQ server
L_i	Packet size of i^{th} flow
$V(t)$	Vitrual time
σ_i^*	Busrt size of the effective flow
ρ_i^*	Long-term average rate of effective flow
$R(t)$	Cumulative arrival function of the flow
$R^*(t)$	Cumulative Arrival function of effective flow
δ	Deadline bound requirements

4.2.1 The Effective Flow Curve

To compute the delay bound of a stream served by a (m,k)-WFQ scheduler, we need to estimate the arrival curve of the *effective flow* transmitted by the server. This curve includes all mandatory packets and the maximum number of optional packets transmitted by the scheduler.

Let us consider a flow *S* with cumulative arrival function *R(t)* upper constrained by the arrival curve $\alpha(t) = \sigma + \rho \cdot t$.

Figure 2 shows the two extreme curves that bound this traffic. The upper curve includes all packets of the incoming flow. The lower curve is the lowest possible and includes only mandatory packets of the stream. The *effective input*, denoted $R^*(t)$, served by the WFQ scheduler has an arrival curve $\alpha^*(t) = \sigma^* + \rho^* \cdot t$ located between these two extreme envelopes.

At first, we notice that if the (m,k)-WFQ scheduler is configured to serve only mandatory packets then all optional packets are dropped, and the effective flow is simply the output from a (m,k)-filter device. Therefore, from *Theorem 3* we have:

$$\begin{cases} \sigma^* = \frac{m}{k} \cdot \sigma \\ \rho^* = \frac{m}{k} \cdot \rho \end{cases}$$

Since the minimum guaranteed rate is ρ, the delay bound is simply:

$$D^*_{\max} = \frac{m}{k} \cdot \frac{\sigma}{\rho} + \frac{L_{\max}}{C} \qquad \textbf{(3)}$$

However, the real system does not discard all optional packets. For this purpose, we must estimate the bound on the number of optional packets that the (m,k)-WFQ scheduler would transmit. From definition, an optional packet is dropped whenever it misses its deadline when it would be served. Denote by δ the required deadline. Given that ρ is the guaranteed rate then the maximum burst size *b* that satisfies this delay is defined as:

$$b = \delta \cdot \rho \qquad \textbf{(4)}$$

Naturally, $b \le \sigma$ since we assume that the required deadline is lower than that guarateed by WFQ. Otherwise, if WFQ guarantees the hard real-time deadline requirement then the (m,k)-firm constraint is also satisfied.

The effective flow model is presented in figure 3.

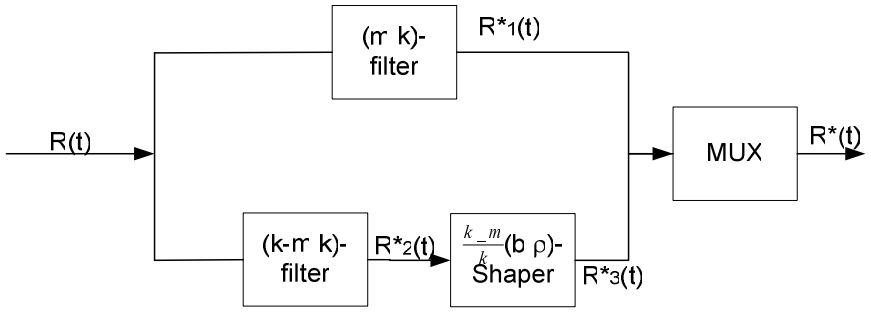


Fig 3. Effective Flow Model

The mandatory part of *R(t)* is the output of the (m,k)-filter and denoted $R_1^*(t)$. The optional part of *R(t)* is obtained when the flow crosses the *(k-m,k)-filter* according to the *reverse κ-pattern* of the stream. The output is denoted by $R_2^*(t)$. Finally, to get the maximum number of optional packets transmitted by the scheduler (not dropped), the flow $R_2^*(t)$ is shaped by a $\frac{k-m}{k}(b,\rho)$ leaky bucket controller to select only optional packets whose deadlines are lower than $\frac{b}{\rho}$.

Then,

- $R_1^*(t) \sim \left(\frac{m}{k}\sigma, \frac{m}{k}\rho\right)$
- $R_2^*(t) \sim \left(\frac{k-m}{k}\sigma, \frac{k-m}{k}\rho\right)$
- $R_3^*(t) \sim \left(\frac{k-m}{k}b, \frac{k-m}{k}\rho\right)$

and the output of the multiplexer $R^*(t)$ defined as:

$$R^*(t) = R_1^*(t) + R_3^*(t)$$

represents the *effective traffic* served by the WFQ server and is bounded by the arrival curve:

$$R^*(t) \sim \left(\frac{m}{k}\sigma + \frac{k-m}{k}b, \rho\right)$$

4.2.2 The Delay Bound

Saying that a flow with an arrival curve $\alpha(t) = \sigma + \rho \cdot t$ is served by a (m,k)-WFQ scheduler is equivalent to say that the effective flow constrained by the arrival curve $\alpha^*(t) = \sigma^* + \rho^* \cdot t$ is served by a standard WFQ scheduler, where:

$$\begin{cases} \sigma^* = \frac{m}{k} \cdot \sigma + \frac{k-m}{k} b \\ \rho^* = \rho \end{cases}$$

Consequently, the delay bound guaranteed by (m,k)-WFQ server for the stream *S* is:

$$D^*_{\max} = \frac{\sigma^*}{\rho} + \frac{L_{\max}}{C} \quad \Rightarrow$$

$$D^*_{\max} = \frac{m}{k} \cdot \frac{\sigma}{\rho} + \frac{k-m}{k} \cdot \frac{b}{\rho} + \frac{L_{\max}}{C} \qquad \textbf{(5)}$$

The first summand indicates the maximum mandatory burst size. The second one represents the amount of maximum burst size of served optional packets. The last summand represents the server latency from the GPS fluid model.

On the other hand, the delay bound guaranteed by the standard WFQ scheduling algorithm is [6]:

$$D^{WFQ}_{\max} = \frac{\sigma}{\rho} + \frac{L_{\max}}{C}$$

We observe that the delay bound guaranteed by the (m,k)-WFQ algorithm is lower than that provided by standard WFQ since the filtered burst size σ^* is lower than the original burst size σ.

Moreover, the burst size of the served optional packets could be adjusted to make guaranteed delay bound for mandatory real-time packets. So, if we assume that the maximum required delay for mandatory packets is D_{req}, then the maximum allowed burst size for optional packets is:

$$b = \frac{k}{k-m}\left(D_{req} - \frac{m}{k} \cdot \frac{\sigma}{\rho} - \frac{L_{\max}}{C} \right)\rho \qquad \textbf{(6)}$$

Consequently, the maximum allowed delay to transmit an optional packet would be:

$$\delta_{opt} = \frac{b}{\rho}$$

The minimum upper bound on delay that could be guranateed for real-time mandatory packets is obtained for $b = 0$ which corresponds to drop all incoming optional packets.

As a result, it can be observed that using (m,k)-WFQ scheduling algorithm is very efficient to smooth the effect of bursty traffic and to guarantee lower delay bound with respect to its (m,k)-firm constraint without violating the throughput fairness. Hence, (m,k)-WFQ scheduling technique provides both delay and bandwidth guarantee for real-time streams that tolerate some deadline misses according to their κ-patterns.

4.2.3 Case of Variable Packet Size

This analysis could be extended to estimate the delay bound guaranteed by (m,k)-WFQ scheduler for a variable packet-size stream. However, there must be a good knowledge on the packet size variation. Hence, the extension is possible if the packet size variation is cyclic or if the amount of mandatory bits is known. We assume that there exists $\lambda_{m,k}$ the ratio of mandatory bits in the window of k consecutive packets. For a constant packet size, $\lambda_{m,k} = \frac{m}{k}$.

A practical example of this assumption is a VBR source that generates MPEG video traffic according to a defined GoP. The Study presented in [14], shows that for a given stream, the proportion of I-frames, P-frames and B-frames are well known within a GoP. So, saying that all B frames, for example, are considered as optional, it is possible to predict the ratio of mandatory part which is the amount of I-frames and P-frames within a GoP.

Therefore, let $\lambda_{m,k}$ be the ratio of mandatory packets into the window of k consecutive packets according to its κ-pattern.

Equation 5 is also applicable in this case where $\lambda_{m,k}$ represents the ratio of mandatory bits of the flow and $\lambda_{k-m,k}$ is the proportion of the optional part. The delay bound is:

$$D^* = \lambda_{m,k} \cdot \frac{\sigma}{\rho} + \lambda_{k-m,k} \cdot \frac{b}{\rho} + \frac{L_{\max}}{C} \qquad \textbf{(7)}$$

However, for a random packet size stream, analysis is no further possible since the mandatory part is also random and completely unknown.

5. Performance Study

In this section, we present the performance evaluation of the (m,k)-WFQ algorithm using OPNET simulator [8].

The network contains two real-time sources (Voice and Video) and a best-effort one (FTP). These streams share a 10 Mbps link according to their bandwidth requirements and are served by a GR server. We assume that all packets have the same size of 1KB (K Byte). This example is very similar to that used in [2] to evaluate the performance of PWFQ.

Table 3. Simulation Parameter

	(m,k)	Rate	Traffic Model	κ-pattern	Required Deadline
Voice	(4,5)	64 kb/s	ON/OFF (500/755/50) ms	MMOMM	10 ms
Video	(3,5)	2Mb/s	Pseudo Periodic ~2Mb/s	MOMMO	4 ms/ 40 ms
FTP	(0,1)	7,936 Mb/s	Pseudo Periodic ~7.936 Mb/s	O	Inf

Voice stream is modeled as ON-OFF source [10]. Specifically, ON and OFF times are exponentially distributed with $1/\mu_{ON} = 500ms$ and $1/\mu_{OFF} = 755ms$. The packet generation period is 50 ms. Then, the average required bandwidth is 64 Kbps. The required delay is 10 ms with a (4,5)-firm constraint. The stream is not shaped by a leaky bucket controller to see clearly the impact of the burst on the delay bound.

Video stream is modeled as a CBR source that generates pseudo-periodic traffic at an average rate of 2 Mbps and a deadline requirement of 4 ms with (3,5)-firm constraint.

The Best-Effort traffic is an aggregate FTP traffic that consumes the remained bandwidth at an average rate of 7.936 Mbps. There is no real-time constraint for FTP traffic, so all FTP packets are considered as optional with (0,1)-firm guarantee. However, We configure (m,k)-WFQ server to drop optional packets only for real-time streams *i.e.* voice and video. All packets of FTP traffic are transmitted since this application is very sensitive to packet loss but not to the delay. We set the FTP deadline to be infinite.

5.1 Delay Guarantees

In this scenario, we also present the delay bound provided by the (m,k)-FIFO scheduling algorithm, that simply behaves as a FIFO server and drops optional packets of real-time streams when they miss their required

deadlines according to their (m,k)-firm timing constraints. The purpose is to compare a best-effort scheduling technique with drop, *i.e.* (m,k)-FIFO with a fair scheduling technique with drop *i.e.* (m,k)-WFQ.

Based on our results in section 4, we deduce the analytic upper bound on the delay for the aggregate of all flows served by a (m,k)-FIFO scheduler is:

$$D_{\max}^{(m,k)-FIFO} = \frac{1}{C}.\sum_{i=1}^{N}\left(\frac{m_i}{k_i}\sigma_i + \frac{k_i - m_i}{k_i}b_i\right) \quad \textbf{(8)}$$

where C is the capacity of the scheduler, (m_i,k_i) is the firm constraint of the stream, σ_i is the maximum burst size and b_i is the allowed burst size for optional packets. N denotes the number of served stream. This upper bound is guaranteed for all served streams.

Table 4 shows the experienced delay bounds of each stream using standard WFQ, (m,k)-WFQ, FIFO and (m,k)-FIFO schedulers.

Table 4. Experienced Delay Bound (ms) (1st Scenrio)

	(m,k)-WFQ	WFQ	FIFO	(m,k)-FIFO
Voice	9.769	2428.031	48.031	5.111
Video	3.999	55.391	49.031	4.952
FTP	9.696	36.562	49.083	5.339

We alert the reader that the WFQ system does not reach a steady state, as it could be understood from Figure 5. So, just for comparative purpose, the experienced delay bounds are made for a large but limited simulation time. These delay bounds are not the absolute bounds, as the system does not reach a steady state. According to queueing theory, a random arrival process system with average load equals to 1, provide delays that tend to infinite [13].

As expected, the (m,k)-WFQ provides shorter delay bound than that guaranteed by WFQ for each stream. In fact, the FTP application is greedy and has the highest service share although it has no critical real-time constraint. Standard WFQ server could not differentiate between streams only according to their *Finish Tag*. For that reason, voice session, which has the lowest service share and the biggest burst size, suffers from an important experienced delay bound.

However, (m,k)-WFQ, by using the information of the timing constraints (deadlines and (m,k)-firm requirements), reduces considerably the measured delay for real-time streams. In this example, since FTP traffic

has no real-time constraint, all of its packets are considered as optional by the (m,k)-WFQ server. Therefore, (m,k)-WFQ scheduler serves mandatory packets of real-time streams before the best-effort traffic. This is not possible using standard WFQ since it cannot differentiate between different kinds of served flows.

On the other hand, (m,k)-FIFO provides a delay bound of about 5 ms for all the stream as there is no service differentiation. As the delay requirement of voice packet is higher than (m,k)-FIFO delay bound, all voice packets are successfully transmitted, whereas 3% of video packets are lost due to dropping optional packets. However, The behavior of (m,k)-FIFO, as well as (m,k)-WFQ, depends a lot on the required deadline for mandatory packets.

In fact, we have changed the deadline value of video stream to 40 ms in the second scenario.

Table 5 shows the maximum delay measured for each stream for standard WFQ, (m,k)-WFQ, FIFO and (m,k)-FIFO schedulers in the second scenario.

Table 5. Experienced Delay Bound (ms) (2nd Scenrio)

	(m,k)-WFQ	WFQ	FIFO	(m,k)-FIFO
Voice	4.507	2428.031	48.031	20.529
Video	39.92	55.391	49.0314	21.086
FTP	32.449	36.562	49.083	21.442

In this scenario, the (m,k)-FIFO delay bound is about 21 ms. Simulation results shows that 52% of voice stream packets have missed their deadlines, which is not acceptable since the mandatory packet ratio is 4/5, whereas all video packets have been successfully transmitted. So, there is no a per-flow delay guarantee with (m,k)-FIFO although there is a noticeable delay reduction compared to FIFO queue.

On the other side, (m,k)-WFQ provides shorter delay bound for voice stream than the first scenario and higher delay bound for video stream, but remains lower than the requested deadline. This result is linked to the drop ratio. Table 6 presents the drop ratios in the case of (m,k)-WFQ and the deadline miss ratio in the case of (m,k)-FIFO for voice and video streams in both scenarios. The drop ratio refers to the ratio of optional packets that have missed their deadlines and dropped by the (m,k)-WFQ server.

Table 6. Drop Ratio / Deadline Miss Ratio

	Drop Ratio (m,k)-WFQ		Deadline Miss Ratio (m,k)-FIFO	
	1er Scenario	**2ème Scenario**	**1er Scenario**	**2ème Scenario**
Voice	5.07%	8.9%	0%	52%
Video	4.21%	0%	3%	0%

We notice that the video packet drop ratio is null in the second scenario. This says that all mandatory and optional packets have been transmitted in time, which explains the expending of FTP delay bound whose packets are optional. In fact, dropping optional packets from video stream in the first scenario makes faster transmission of the FTP traffic.

However, the improvement seen on the voice delay bound is due to the increase of the drop ratio from 5.07% to 8.9%, but the (4,5)-firm constraint is still satisfied in both scenarios since the guaranteed delay bound is lower than requested.

We mention that in both scenarios, all mandatory packets have met their required deadlines when they are served by (m,k)-WFQ which is not the case for the other scheduling techniques. Thus, as shown analytically, (m,k)-WFQ provides a per-flow delay as well as bandwidth guarantee with respect to the (m,k)-firm constraints.

Figures 4 and 5 depict the average queueing delay experienced by each stream when scheduled by (m,k)-WFQ server and standard WFQ server, respectively.

We notice that the bursty behavior of the voice source has a negative effect on the queueing time into a WFQ server, whereas when taking into account the (m,k)-guarantees, average delays are in steady state.

5.2 Resource Utilization

For a Service Provider (SP), resource utilization is an important issue to grant en efficient resource share for all guaranteed service flows.

From this scenario, assume that the service provider uses fair queueing scheduler to make guaranteed service without taking into account the (m,k)-firm temporal constraints and assume that the required QoS for voice stream consists in a short delay no more than 10 ms. From results in table 4 of WFQ scheduler, it is not sufficient that the SP gives a service share proportional to the average rate of the stream *i.e.*, 64 kbps, although it is the actual need of the voice application. Hence, the Service

Provider has to make higher service share to the voice application in order to grant the required temporal QoS.

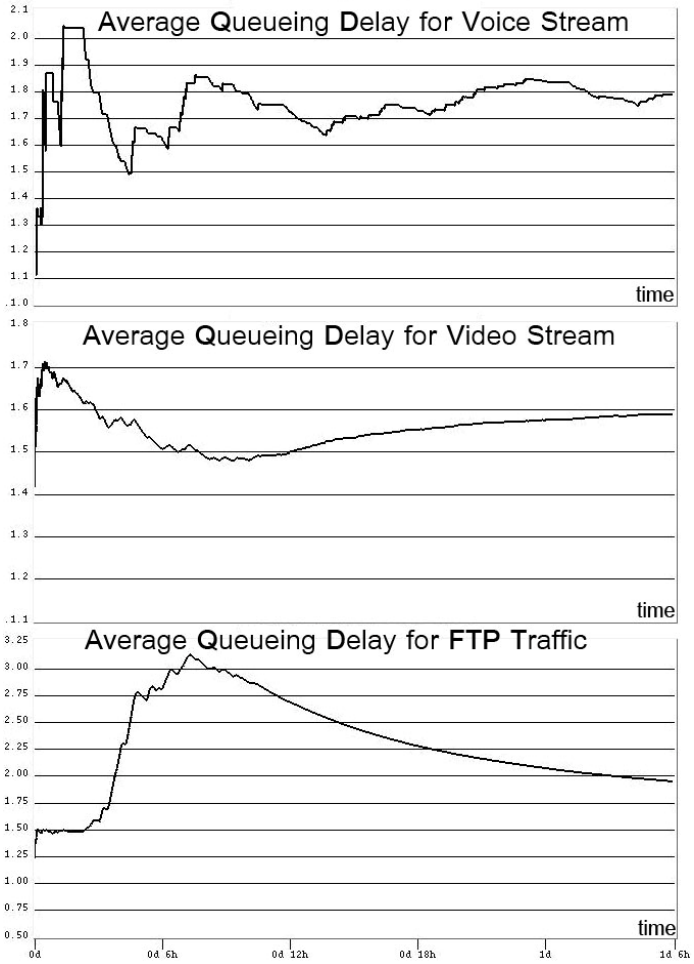


Fig 4. (m,k)-WFQ Scheduler Average Queueing Delay (ms)

The first drawback of this solution is economical. In fact, the user should pay the cost of an additional bandwidth to get the required temporal QoS.

Secondly, this solution leads to misuse the bandwidth resources and the link may not be efficiently utilized which results in the admission of fewer number of streams. In fact, simulation shows that for the 64kbps-voice traffic, the service share should be 96 kbps to make guaranteed delay lower than 10 ms with a 50% additional bandwidth. This solution, which consists in decreasing the service share of FTP traffic, leads to

violate the normal share of this flow. Throughput fairness is no longer satisfied.

However, this problem can be resolved by integrating (m,k)-firm timing constraints, to make both bandwidth and delay guarantees. The service provider could adjust the maximum deadline to drop optional packet, using equation 6, in order to control the guaranteed delay for real-time packets without violating the QoS of other real-time streams, and conserve the fairness of bandwidth reservation for better resource management.

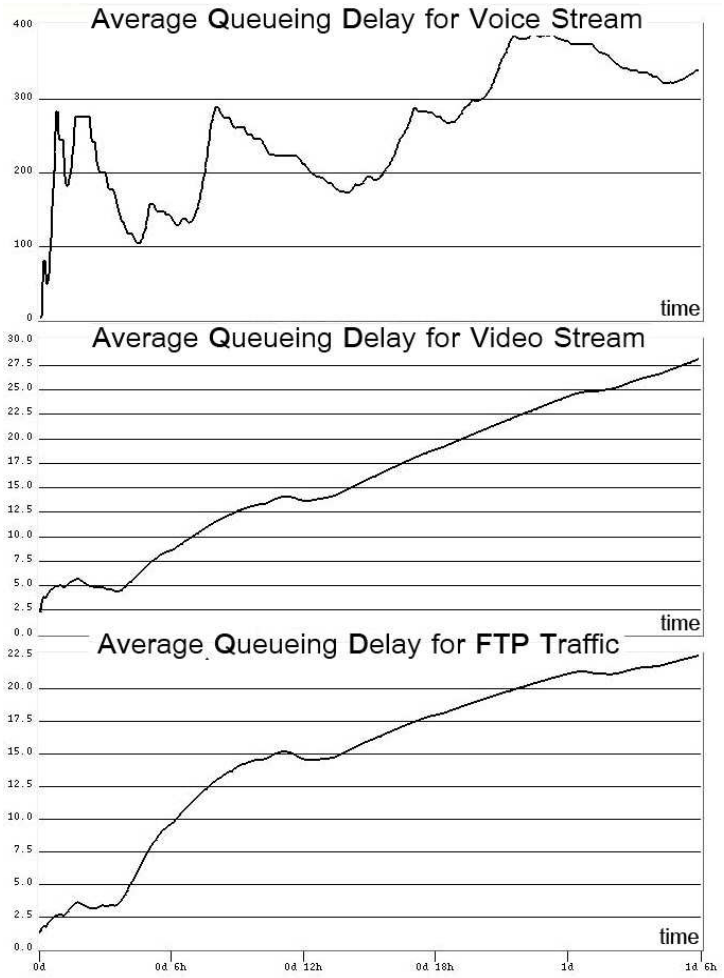


Fig 5. WFQ Scheduler Average Queueing Delay (ms)

6. Conclusion

In this paper we have proposed a new fair scheduling algorithm, called (m,k)-WFQ, that enhances fair queueing algorithm in terms of delay guarantee by taking account of (m,k)-firm timing constraints. For this purpose, we have proposed a marking process for incoming packets as mandatory or optional according to their (m,k)-firm constraints and their κ-patterns. Also, we have introduced the concept of (m,k)-filtering to extend analytic results on delay bound using network calculus framework. Simulation scenario approved analytic results and show better behavior of (m,k)-WFQ versus WFQ en terms of delay guarantee and resource management.

The main advantage we have noticed is that (m,k)-WFQ provides a well-defined delay guarantee without violating the throughput fairness. This fact results from the integration of (m,k)-firm requirements into the fair scheduling process. Packet drop policy used by (m,k)-WFQ server makes shorter delay bounds as optional bursty packets are removed if they miss their required deadline.

We showed that (m,k)-WFQ technique distinguishes between real-time flows and best effort-ones. This is a very interesting issue as low service share stream could have lower delay bound without need to get larger bandwidth reservation.

We are working towards to integrate (m,k)-firm constraints into DiffServ and IntServ QoS architecture . Also, a dynamic classification of packet priority into mandatory and optional classes to have optimal packet drops is in progress.

7. References

[1] S. Floyd, V. Jacobson "Random Early Detection for Congestion Avoidance" *IEEE/ACM Transactions on Networking* V.1 N.4, August 1993, p. 397-413.

[2] S. Wang, Y. Wang, K. Lin, "Integrating Priority with Share in the Priority-Based Weighted Fair Queueing Scheduler for Real-Time Networks*" Journal of Real Time Systems*, pp. 119-149, Vol. 22, 2002.

[3] M. Hamdaoui, P. Ramanathan. "A dynamic priority assignment technique for streams with (m, k)-firm deadlines*" IEEE Transactions on Computers*, 44 (4), 1443–1451, Dec.1995.

[4] G. Bernat, A. Burns, A.Lamosi "Weakly Hard Real-Time Systems", *IEEE Transactions on Computers*, 50 (4), pp.308-321, 2001.

[5] P. Ramanathan "Overload Management in Real-Time Control Applications Using (m,k)-Firm Guarantees" *IEEE Transactions on Parallel and Distributed Systems*, Vol. 10, No. 6, pp. 549-559, June 1999.

[6] J.Y. LeBoudec, P. Thiran, 'Network Calculus : A Theory of Deterministic Queueing Systems for the Internet' Springer Verlag, July 2002.

[7] A. Koubâa, Y.Q Song "Evaluation et amélioration des bornes de temps de réponse pour des applications temps-réel avec ordonnancement à priorité fixe et non-préemptif " 4ème Colloque Francophone sur la Modélisation des Systèmes Réactifs MSR'03, pp. 127-144 Metz, Octobre 2003.

[8] R.L. Cruz. A calculus for network delay, part II: Network analysis. *IEEE Transactions on Information Theory*, 37(1): 132-141, Jan. 1991.

[9] R.L. Cruz. A calculus for network delay, Part I. *IEEE Transactions on Information Theory*, 37(1):114-131, Jan. 1991.

[10] K. Sriram and W. Whitt, "Characterizing superposition arrival processes in packet multiplexers for voice and data," *IEEE J. Select. Areas Commun*., vol. SAC-4, pp. 833--846, Sept. 1986.

[11] A. Koubâa, A. Jarraya, Y.Q. Song "SBM protocol for providing real-time QoS in Ethernet LANs", *1st Intl. Workshop on Real-Time LANs in the Internet Age* (Satellite Event to ECRTS'02), Austria, Juin 2002.

[12] A. Demers, S.Keshavt, S.Shenker, "Analysis and Simulation of Fair Queuing Algorithm*", Proceedings ACM SigComm* 89, pp. 3-12, 1989.

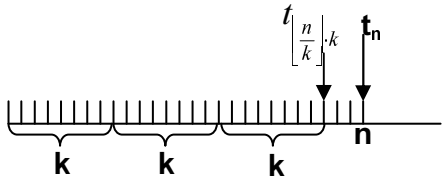
[13] A. Allen, 'Probability, Statistics and Queueing Theory with Computer Science Applications' Second Edition, Academic Press 1990.

[14] J.M. Boyce, R.D. Gaglianello, "Packet Loss Effects on MPEG Video Sent Over the Public Internet*" Proceedings of the 6th ACM International Conference on Multimedia* September 1998.

[15] J.C. Bolot, H. Crépin, A.V. Garcia, *"*Analysis of Audio Packet Loss in the Internet*" Proceedings Networks and Operating System Support for Digital Audio and Video*, pp. 163-174, Durham, NH, April 1995.

Appendix 1 – Proof of Theorem 1

$\Rightarrow$



We denote by t_n the instant of n^{th} packet arrival. $\forall n, \forall t_n \le t < t_{n+1}, \quad R(t) = n$.

- ***If n is multiple of k***: then it exists u such that $n = u \cdot k$. In this case, there are u consecutive and separate sets of k consecutive packets. Since only m packets out of any k packets exit the filtering device toward the network, then there is exactly

$m \cdot u$ mandatory packets that leave the *(m,k)-filter*. So, $\left\lfloor \frac{R(t)}{k} \right\rfloor = u$ and

$$\Pi(R(t)) = \sum_{n=(u.k)+1}^{u.k} \overline{\kappa}(n \bmod k) = 0$$

and *Theorem 1* is satisfied since $\widetilde{R}(t) = m.u$

- ***If n is not a multiple of k***: it exists $u \in \mathbb{N};\ \frac{n}{k} - 1 \le u \le \frac{n}{k}$ and $v \in \mathbb{N};\ 1 \le v \le k-1$ such that $n = u \cdot k + v$.

  By definition, $u = \left\lfloor \frac{n}{k} \right\rfloor$ and $v = n - u.k$. At time $t_{u.k}$ there are $m \cdot u$ packets that leave the *(m,k)-filter*. From $t_{u.k}$ until t_n the filter sends only mandatory packets according to the defined κ-pattern κ. From *Proposition 1,* the n^{th} packet is mandatory if and only if $\overline{\kappa}((n \bmod k)+1) = 1$, in the time interval $[t_{u.k}, t_n]$, the number of mandatory packets is $\sum_{a=(u.k)+1}^{(u.k)+v} \kappa(n \bmod k) \in [0, m]$. Then ,

$$\widetilde{R}(t) = m \cdot u + \sum_{a=(u.k)+1}^{(u.k)+v} \overline{\kappa}(n \bmod k)$$

  which satisfies *theorem 1*.

  $\Leftarrow$ Evident.

Appendix 2 – Proof of Theorem 2

$\Rightarrow$

The proof is evident. In fact

$\Lambda(t) = \sum_{n=1}^{N^-(t)} \overline{\kappa}(n \bmod k).L_n$ is the number of bits of all mandatory packets completely received, at time t, that cross the (m,k)-filter.

$\overline{\kappa}(N^+(t) \bmod k).\left(R(t) - \sum_{n=1}^{N^-(t)} L_n\right)$ is null if the current received packet is optional. Else, it is equal to the number of received bits, at time t, from the current packet.

Appendix 3 – Proof of Theorem 3

Let us consider a constant-packet size stream with cumulative arrival function *R(t)* shaped with a leaky bucket controller with σ as maximum allowed burst and ρ as the average long-term rate. Then,

$$R(t) - R(s) \le \sigma + \rho(t-s) \ , \qquad \forall\ 0 \le s \le t$$

Also, we assume that this stream has (m,k)-firm constraints given by its κ-pattern κ. What kind of arrival curve would have the stream after crossing an (m,k)-firm filter is the purpose of this proof.

The output $\widetilde{R}(t)$ of the stream satisfies Theorem 1. So, $\forall\ 0 \le s \le t$

$$\widetilde{R}(t) - \widetilde{R}(s) = m \cdot \left\{ \left\lfloor \frac{R(t)}{k} \right\rfloor - \left\lfloor \frac{R(s)}{k} \right\rfloor \right\} + \left\{ \Pi\left(R(t)\right) - \Pi\left(R(s)\right) \right\}$$

We aim to determine an arrival curve at the output of the (m,k)-filter, which is independent from any κ-pattern. However, the function $\pi(t-s)$ defined as:

$$\pi(t-s) = \left\{ \Pi\left(R(t)\right) - \Pi\left(R(s)\right) \right\}$$

is non-increasing and non-monotonic function and depends on κ-pattern setting of the stream. For this reason, we consider a new discrete sampling time $\mathrm{T} = \{t_0, t_k, t_{2k}, ..., t_{nk}, ...\}$ where t_{nk} represents the *(nk)*th packet arrival time. According to [6], this kind of mapping results in some loss of information but it is sufficient to get an arrival curve for $\widetilde{R}(t)$. With the new sampling time, $\forall\ s,t\ \in T,\ 0 \le s \le t;\ \ \pi(t-s) = 0$ and:

$$\left\lfloor \frac{R(t)}{k} \right\rfloor = \frac{R(t)}{k};\ \ \forall t \in \mathrm{T}\ .$$

Then,

$$\forall s,t \in \mathrm{T},\ s \le t,$$

$$\widetilde{R}(t) - \widetilde{R}(s) = m \cdot \left(\frac{R(t)}{k} - \frac{R(s)}{k} \right)$$

So, the output flow is constrained as follows:

$$\widetilde{R}(t) - \widetilde{R}(s) \le \frac{m}{k} \cdot \left(\sigma + \rho \cdot t \right)$$